ROBOFÁCIL – KIT DE ROBÓTICA EDUCACIONAL REPROGRAMÁVEL: DESIGN E IMPLEMENTAÇÃO

LEONARDO CUNHA de Miranda
IM-NCE/UFRJ – FAETEC – IPLANRIO
professor@leonardocunha.com.br

José ANTONIO dos Santos BORGES
NCE/UFRJ
antonio2@nce.ufrj.br

Fábio FERRENTINI SAMPAIO
NCE/UFRJ
ffs@nce.ufrj.br

RESUMO

O presente trabalho apresenta o Kit de Robótica Educacional Reprogramável denominado RoboFácil, em desenvolvimento pelo Grupo de INformática APlicada a Educação (GINAPE) do NCE/UFRJ.

O RoboFácil tem como objetivo principal apoiar o ensino de ciências em sala de aula, criando novos paradigmas, tanto para os discentes como para os docentes, e tornar a robótica educacional uma realidade viável no contexto escolar brasileiro.

Palavras-chaves: robótica, informática e educação, eletrônica.

1 – INTRODUÇÃO

A robótica educacional desperta no aluno o gosto pelas ciências e por novas tecnologias, trabalhando com conceitos de matemática, física, informática, eletrônica, mecânica e arquitetura, possibilitando também o desenvolvimento da lógica, da organização e do planejamento através da elaboração de programas que darão vida aos modelos criados [6].

A metodologia aplicada à robótica educacional deve centrar-se na implementação de ambientes de aprendizagem ricos em situações que permitam ao aluno construir o seu conhecimento, através do uso do microcomputador e dos dispositivos robóticos. Tal metodologia deve propiciar subsídios para uma diversificação, diferenciação e expansão na forma de aquisição, assim como, no manuseio de conceitos [8].

Diversos experimentos com robótica vêm sendo realizados em diferentes Universidades no mundo [7,15,28,33]. No Brasil, podemos citar os projetos em desenvolvimento no Laboratório de Estudos Cognitivos da Universidade Federal do Rio Grande do Sul (LEC/UFRGS) [29] e no Núcleo de Informática Aplicada a Educação da Universidade de Campinas (NIED/UNICAMP) [31].

Duas características comuns à maioria dos projetos existentes nas escolas, são fatores que contribuem para a baixa expansão da utilização da robótica educacional no país: a primeira delas é o fato de utilizarem kits importados – como o LEGO MINDSTORMS [16] – que apresenta um alto custo de aquisição (cerca de US$ 300 por kit); a segunda é a limitação técnica dos produtos desenvolvidos para o mercado nacional, que impedem o desenvolvimento de diversos projetos pedagógicos em sala de aula.

Em 2001, José Antonio dos Santos Borges e José Henrique Gandra, respectivamente alunos do Curso de Doutorado e Mestrado do IM-NCE/UFRJ, desenvolveram como projeto final da disciplina Introdução à Informática na Educação lecionada pelo Prof. Fábio Ferrentini Sampaio [6], a base para a construção de um kit de robótica educacional. Esse trabalho consistiu na especificação e montagem de um *hardware* – base de todo o projeto, composto por componentes eletrônicos associados – e alguns programas (desenvolvidos na linguagem *assembly)*, para controlar algumas funções básicas desse equipamento[1].

Na primeira fase do projeto, apenas algumas implementações foram realizadas, com o intuito de testar o funcionamento do *hardware*. Na segunda fase do trabalho, objeto deste Relatório, foi proposto então a conclusão do projeto, com a construção de todos os *plugins*[2] e o desenvolvimento de *software* para controlar o Kit, com os quais os alunos poderiam expressar-se "facilmente" na criação de modelos e projetos próprios.

No início da 2.ª fase do projeto foi necessária uma manutenção inicial do protótipo, visto que o mesmo se encontrava em desuso há cerca de dois anos.

---

[1] A arquitetura eletrônica desse hardware foi projetado pelo M.Sc. Eng. Diogo Fujio Takano.

[2] Nesse projeto, um *plugin* eletrônico pode ser definido, como um *hardware* externo ao circuito principal, que possui a flexibilidade de ser acoplado facilmente ao sistema sem que haja necessidade de alteração no projeto eletrônico para utilizá-lo.

Para atender as especificações levantadas, foram realizadas algumas alterações no projeto eletrônico, com o intuito de otimizar a utilização de componentes eletrônicos, como também, facilitar a execução, manutenção e depuração de eventuais panes durante a utilização do Kit de Robótica. Especificamente, foram montados quatro *plugins* com diferentes componentes eletrônicos (resistores, capacitores, diodos, transistores, circuitos integrados, LDR, NTC, motores de passo, etc.) e algumas rotinas básicas em linguagem C para controlar os dispositivos eletrônicos contidos no Kit, agora batizado de RoboFácil.

Nos próximos capítulos serão apresentados as características eletrônicas da arquitetura adotada no RoboFácil, como também o desenvolvimento do *software* básico que controla o Kit. Os diagramas esquemáticos atualizados e a versão atual do *software* básico do Kit de Robótica Educacional encontram-se nos Anexos I e II.

2 – O ROBOFÁCIL E O LEGO MINDSTORMS

A proposta foi desenvolver um kit de robótica educacional usando a mesma filosofia utilizada pela LEGO [17] em parceria com o Massachusetts Institute of Technology (MIT) para desenvolver o LEGO MINDSTORMS [19,16]. No caso da LEGO, o Robotic Command eXplorer (RCX), que possui um microprocessador da série Hitachi H8/3292, é o principal componente, podendo ser considerado o "cérebro" do Kit. No Kit de Robótica Educacional em desenvolvimento pelo GINAPE/NCE, essa função é executada pelo microcontrolador Intel® 8031AH.

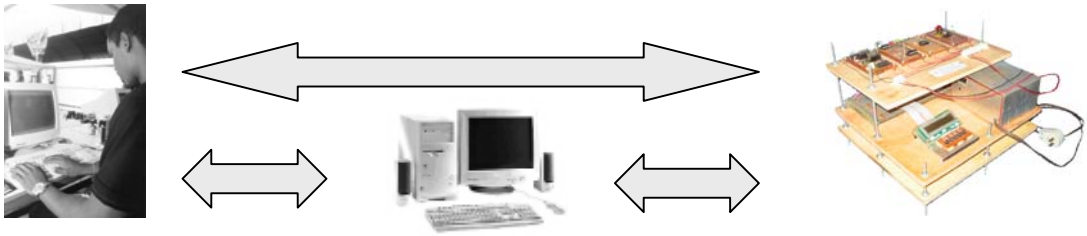
Podemos citar pelo menos duas grandes vantagens na utilização dessas CPUs nos kits de robótica supracitados. A primeira delas, reside na possibilidade dos mesmos serem reprogramados, permitindo, dessa forma, que os alunos possam produzir uma infinidade de comportamentos para os kits, possibilitando sua utilização em diversos projetos pedagógicos interdisciplinares.

A segunda vantagem é a característica dos kits poderem ser desconectados dos microcomputadores após terem seu comportamento estabelecido (programado) pelos alunos. Essa propriedade torna esses kits extremamente versáteis para serem utilizados em projetos educacionais.

Apesar de ambos os kits de robótica possuírem flexibilidades semelhantes, o alto custo do LEGO MINDSTORMS inviabiliza o seu uso em projetos educacionais em escolas brasileiras.

A criação dos projetos de robótica pelos alunos utilizando o kit de robótica educacional do GINAPE segue os seguintes passos (vide fig. 1): (1) construção dos modelos utilizando peças

de sucata ou do tipo LEGO e os componentes eletrônicos que fazem parte do Kit; (2) criação no microcomputador do programa que irá determinar o comportamento do modelo construído; (3) transferência do programa criado para o Kit, utilizando a *interface* serial do microcomputador; (4) após a carga do programa, o Kit de robótica pode ser desconectado do microcomputador para iniciar sua execução e interação com o aluno.



[Aluno] [Microcomputador] [Kit de Robótica Educacional]

Figura 1 – Esquema de utilização do Kit de Robótica Educacional RoboFácil

3 – ESPECIFICAÇÃO ELETRÔNICA

Estaremos aqui apresentando a especificação de todos os elementos de *hardware* do ambiente.

3.1 – ARQUITETURA GERAL

O Kit de Robótica Educacional RoboFácil é baseado em uma CPU da família dos microcontroladores MCS-51 da Intel® e contém uma extensão de 16 portas digitais de entrada e 16 portas de saída, com bufferização.

O sistema provê a utilização de circuitos externos – denominados *plugins* – que permitem a interação do sistema com o mundo externo. Um plugin acoplado ao kit pode assim controlar, por exemplo, um motor de passo, um sensor de temperatura ou luminosidade.

Os principais elementos do Kit de Robótica são:

- CPU – Intel® 8031AH (CPU CMOS de baixo consumo);
- 32 KB de Memória ROM de programa e 32 KB de Memória RAM de programa;
- 8 KB de Memória RAM de dados;
- *Interface* de comunicação RS-232C utilizando conector externo DB-25;
- Mostrador (display) de 01 (uma) linha com 16 (dezeseis) colunas;
- Teclado (botoeira) com 05 (cinco) botões;
- Controle de 02 (dois) motores de passo;

- Controle de motores DC (apenas projetado);
- 01 (um) sensor de temperatura;
- 01 (um) sensor de luminosidade;
- 01 (um) conversor digital-analógico;
- 01 (um) conversor analógico-digital.

3.2 – UNIDADE DE CONTROLE E SISTEMA DE MEMÓRIA

As características da unidade de controle derivam-se das propriedades operacionais das CPUs da linha Intel® 8051, já projetada visando a implementação de sistemas de controle de equipamentos por microprocessador, fazendo uso de um *hardware* mínimo. Existem diversos modelos dessas CPUs e cada um podendo ser configurado de diferentes formas[3].

A CPU 8051 é composta por três conjuntos de portas externas bidirecionais de 8 bits (P0, P1 e P2), endereçáveis bit a bit. Alguns modelos dessa CPU podem ser utilizados com ROM e RAM externas, e nesse caso, duas portas (P0 e P2) são usadas para controle dos barramentos externos de dados e endereços. Ao implementar o barramento, as portas P0 e P2 são concatenadas para formar um endereço de 16 bits (dando assim o máximo de 2^{16}=64 KBytes de acesso externo). A porta P0 deve ser multiplexada através de um circuito 74HCT373, para fornecer o barramento bidirecional de dados (vide Anexo I.1).

A CPU implementa internamente um serializador/desserializador, que serve para realizar a conexão serial do processador com o mundo externo. É possível configurá-lo para comunicação síncrona e assíncrona (embora nesse trabalho tenha sido utilizado apenas a forma assíncrona). A velocidade de comunicação é dada por um *timer* interno, que, no caso da configuração de *hardware* desse projeto, permite conexão de até 2.400 bps (com a mudança do cristal utilizado, pode-se ir até 9.600 bps).

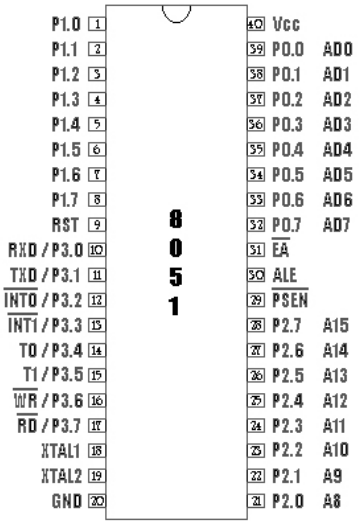


Figura 2 – Pinagem do 8051

A CPU possui sinais diferentes para leitura de programa e leitura de dados, aumentando teoricamente a quantidade possível de memória. Esse fato foi aproveitado pelo projeto, servindo-se de uma área de ROM para rotinas básicas (32KB), uma área de RAM para programa (32KB) e

[3] Uma discussão completa sobre as múltiplas configurações da linha 8051 foge ao escopo desse trabalho. Podendo ser consultado em: NICOLOSI, Denys C. **Microcontrolador 8051 Detalhado**. Ed. Érica. 221 p. ISBN: 857194721

uma área de RAM para dados (8KB). Dessa forma, simplifica-se o processo de testagem de novas rotinas básicas, carregando-as, via *interface* serial, na área de RAM de programa – o "bootstrap" é feito a partir das rotinas já existentes em ROM. Numa próxima etapa do projeto, quando tais rotinas já estiverem suficientemente testadas, poderão então ser gravadas em ROM, eliminando-se assim, a necessidade do uso dessa RAM de programa.

Devido ao grande número de portas utilizadas nesse projeto, foi escolhido o modelo de endereçamento de portas no espaço em memória. Para os programas, as diversas *interfaces* externas são lidas e gravadas como posições de memória. Dois circuitos NAND (74HCT00) e dois registradores de 4 bits (74HCT139) controlam e selecionam esses endereços e suas incompatibilidades com o espaço real de memória (vide Anexo I.1).

3.3 – SUBSISTEMAS INTERNOS

3.3.1 – COMUNICAÇÕES

Os sinais de comunicação serial da CPU são TXD e RXD (*Transmitter* e *Receiver Data*), que serão ligados ao exterior através de uma conversão de nível e corrente, realizada por um circuito MC145407. Esse circuito permite, na verdade, a interconexão de seis sinais, e assim, aproveitamos para gerar quatro outros sinais que podem ser utilizados, por exemplo, para controle de um modem externo (permitindo até mesmo a conexão futura do RoboFácil à Internet). Esses sinais extras são lidos numa porta específica, a partir de um *latch* 74HCT244 (vide Anexos I.1 e I.3).

Os sinais gerados por essa placa são levados a um conector externo de 25 pinos (DB25), preparado para conexão a um cabo serial (vide Anexo I.7), que por sua vez é conectado a um microcomputador, permitindo a carga e descarga de dados e programas[4].

3.3.2 – TECLADO (BOTOEIRA)

Foram implementados cinco botões, sendo quatro conectados a um registrador 74HCT244 e um ligado diretamente ao sinal INT0 da CPU (vide fig. 3 e Anexos I.1 e I.3). Esse último foi pensado como uma forma dos programas serem interrompidos externamente, possivelmente numa aplicação de "Stop/Load/Run" de uma ROM de carga.

[4] Nessa versão do RoboFácil ainda não foi implementada a *interface* infra-vermelha para conexão com o microcomputador, apesar de tal característica estar prevista na concepção do *hardware*.

3.3.3 – MOSTRADOR (DISPLAY)[5]

Foi utilizado um *display* programável ALFACOM [24] de uma linha com dezesseis caracteres, modelo LCM 1601 0630 (vide fig. 3). Esse *display* foi escolhido por sua conexão e programação serem extremamente simples. Na realidade, todo processo de controle da varredura e da memória interna é feita por comandos de alto nível, enviados ao *display*, segundo uma tabela fornecida pelo fabricante [24].

Figura 3 – Display e Teclado

A conexão desse *display* é feita diretamente ao *latch* de dados e o endereçamento pelo seletor, da forma mais simples possível[6], pois segue a mesma arquitetura empregada na implementação da conexão dos *plugins* nas portas de I/O do RoboFácil.

3.3.4 – CONVERSOR DIGITAL-ANALÓGICO (D/A)

O RoboFácil provê um conversor digital-analógico, implementado através de um circuito convencional R-2R, construído sobre um conjunto de resistências. O valor obtido está na faixa de 0 à 4 volts, aproximadamente. A entrada do circuito é um registrador 74HC374, que provê razoável isolamento elétrico (vide Anexo I.4).

Apesar de eletronicamente implementado, o conversor D/A não se encontra em utilização nessa versão do Kit, todavia nada impede que seja criada uma aplicação para esse circuito, tal como a síntese de voz.

3.3.5 – CONVERSOR ANALÓGICO-DIGITAL (A/D)

Para realizar a conversão de sinal analógico para digital, optou-se por não fazer uso de nenhum circuito de conversão rápida, e sim realizar todo controle por comparação de nível. A razão para tal é de fácil entendimento: para medir um nível de entrada analógica, gera-se um valor de tensão no conversor digital-analógico, e compara-se o valor obtido com o valor de entrada (através de um comparador analógico construído com amplificadores operacionais

[5] Por simplicidade, não foi implementada a leitura de dados do *display*, que se julgou não ser importante para esse projeto. Também foi mantida fixa (não programável) a luminosidade, ajustada apenas por um potenciômetro interno à placa.
[6] Maiores informações sobre o endereçamento do visor e a tabela de códigos de formação dos caracteres podem ser obtidas no Manual de Utilização dos Módulos Multi-Matrix ALFACOM [24].

LM324 e resistências). Por um processo de busca binária, no máximo em nove comparações, chega-se ao valor da voltagem (valor digital).

Devido à existência de um circuito conversor A/D no RoboFácil, torna-se possível que o Kit de Robótica Educacional possa ser controlado, em futuras versões, por comandos de voz.

3.4 – A INTERFACE GENÉRICA DE PLUGINS

3.4.1 – TÉCNICAS PARA CONEXÃO DE INTERFACES PLUGINS

Um dos problemas mais complexos encontrados na construção de sistemas robóticos é definir exatamente quais serão as *interfaces* utilizadas. Colocando poucas *interfaces*, o sistema fica restrito; se muitas, o sistema fica oneroso e grande. Para solucionar esse impasse foi utilizado no RoboFácil um esquema baseado em *plugins*.

São dezesseis sinais de saída e dezesseis de entrada, nesse esquema, isolados por registradores (74HCT374 e 74HCT244) que podem suportar correntes relativamente altas, protegendo, razoavelmente, o exterior da placa principal do RoboFácil (vide Anexo I.2). Nessas entradas e saídas, ligadas a dois conjuntos de conectores, se acoplarão os diversos circuitos – *plugins* (vide item 3.7).

A dificuldade que essa solução introduz é a necessidade de uma programação cuidadosa e dependente das conexões realizadas.

Nesse projeto, quatro *plugins* foram implementados:

- controle dos motores de passo;
- controle de leds;
- controle do sensor de temperatura;
- controle do sensor de luminosidade.

3.4.2 – PLUGIN DE CONTROLE DOS MOTORES DE PASSO

O circuito de controle dos motores de passo necessita de chaveadores [25] para cada um dos quatro fios de controle do motor, conectando-os a zero ou a um valor entre 9 e 12 V, dependendo do tipo de motor escolhido. A maioria dos motores utiliza uma tabela de códigos que informa os valores de programação como mostrada a seguir:

- 0000: Repouso (motor solto);
- 1000: Motor na posição 1;

- 0110: Motor na posição 2;
- 0011: Motor na posição 3;
- 1001: Motor na posição 4;
- Qualquer outro valor: Instabilidade.

Para movimentá-lo no sentido horário, a seqüência contínua será a posição 1, 2, 3, 4, 1, 2, 3, 4, etc., e em sentido inverso, basta aplicar a seqüência inversa. O tempo de aplicação, histerese[7] e aceleração permitida devem ser obtidos experimentalmente para cada motor (o manual do fabricante apresenta valores esperados e/ou máximos para essas operações). A lógica para o acionamento é totalmente realizada por *software*. O ângulo rodado em cada seqüência também é especificado no manual do motor.

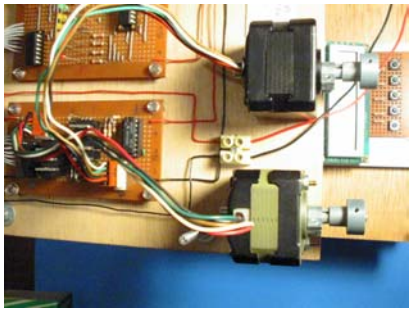


Figura 4 – Dois motores de passo conectados ao plugin

A conexão dos motores foi facilitada com o uso de conectores de cinco pinos, não existindo, portanto, a necessidade dos mesmos serem soldados diretamente na placa de circuito impresso do *plugin* (vide fig. 4). Essa característica facilitará a ligação desses motores, em futuras versões, no primeiro andar do RoboFácil, podendo movimentar, efetivamente, o Kit.

3.4.3 – PLUGIN DE CONTROLE DE MOTORES DC

Nessa versão do RoboFácil, fez-se a opção por não implementar o controle de motores DC visto que os motores de passo apresentam mais vantagens pedagógicas (ex.: controle preciso de velocidade, direção e distância, etc.).

3.4.4 – PLUGIN DE CONTROLE DE LÂMPADAS, RELÉS OU LEDS

O circuito implementado permite a conexão tipo liga/desliga, o que pode ser realizado através de uma chave analógica. Entretanto, nesta versão do kit, por razões pedagógicas, acoplou-se uma barra com oito leds (2 vermelhos, 3 amarelos e 3 verdes), o que permite simular, por exemplo, aplicações educacionais que façam uso de um semáforo (vide fig. 5).

[7] Histerese é uma diferença entre o ponto de acionamento e desligamento, sendo necessária para evitar que o sinal de saída fique "trepidando", ou seja: se não houver histerese, o sinal ficará oscilando quando o sinal monitorado estiver exatamente no valor pré-ajustado.

3.4.5 – PLUGIN DE CONTROLE DO SENSOR DE TEMPERATURA

O sensor de temperatura é binário (ativa/desativa a partir de um determinado valor). Através de um resistor variável – trimpot –, é indicado o valor de referência, sem implementação de histerese. Assim, para temperaturas exatamente iguais ao valor de referência, o circuito pode oscilar, sendo importante procedimentos de *software* para diminuir esse impacto.

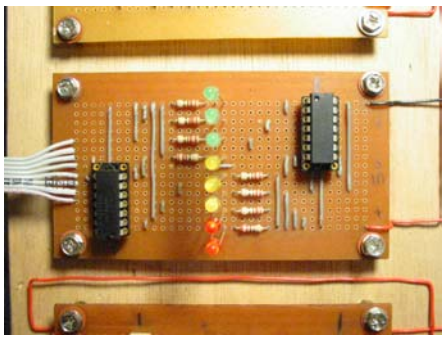


Figura 5 – Plugin de controle dos leds

3.4.6 – PLUGIN DE CONTROLE DO SENSOR DE LUMINOSIDADE

O sensor de luminosidade também é binário, e através de um resistor variável – trimpot – é indicado o valor de referência. O circuito pode oscilar, para intensidades de luz exatamente iguais ao valor de referência, pois o circuito não implementa histerese, visto que, geralmente, o impacto é desprezível para a maior parte das aplicações educacionais que poderão ser desenvolvidas com o Kit.

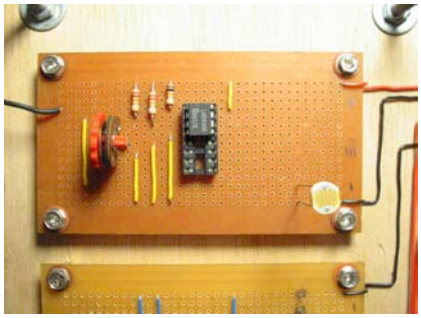


Figura 6 – Plugin de controle do sensor de luminosidade

3.5 – ENDEREÇAMENTOS

É extremamente importante, para os desenvolvedores do Kit de Robótica, conhecer as equivalências de endereços empregadas nesse Kit, pois somente através delas, é possível utilizar os diversos recursos disponíveis no RoboFácil, tais como o *display* e os motores de passo. Contudo, no estágio atual do projeto, para os usuários-alunos, esses endereços não são imprescindíveis, pois os endereçamentos que controlam os diversos recursos já estão mascarados em bibliotecas desenvolvidas na linguagem C.

Apresenta-se abaixo esses endereços:

Memória: (vide Anexo I.1)

PROM Principal	0000h a 7FFFh
RAM Principal	0000h a 1FFFh

RAM / ROM 8000h a FFFFh

I/O: (vide Anexos I.1 a I.3)

Display (Instruções)	6000h (somente escrita)
Display (Dados)	6001h (somente escrita)
I/O 1	6002h (8 bits externos de mais baixa ordem)
I/O 2	6004h (8 bits externos de mais alta ordem)
I/O 3	6006h
b0..b3	(botões)
b4	(DSR)
b5	(CTS)
b6	(valor analógico maior do que o digital gerado)
b7	(não usado)
RTS	bit 0 de P1
DTR	bit 1 de P1
LDR	bit 2 de P1
NTC	bit 3 de P1

3.6 – ALIMENTAÇÃO ELÉTRICA

Todo o circuito do *hardware* de controle, inclusive o circuito RS-232[8], foi preparado para ser alimentado por uma fonte de 5 volts simples. Opcionalmente é possível alimentá-lo com uma fonte um pouco maior (6 volts = 4 pilhas de lanterna), mas não maior que isso, com possibilidade de causar danos irreversíveis.

Havendo a necessidade de fazer uso de uma fonte de alimentação maior, ou uma fonte não regulada, será imprescindível introduzir no projeto um circuito regulador de voltagem convencional. Nesse caso, haverá desperdício de energia no regulador, e sendo consumida mais potência, haverá diminuição da vida útil das pilhas.

Na versão atual do RoboFácil, os motores de passo estão sendo alimentados por uma fonte externa ao Kit. Essa fonte deve fornecer tensões da ordem de 9 a 12 volts para produzir o

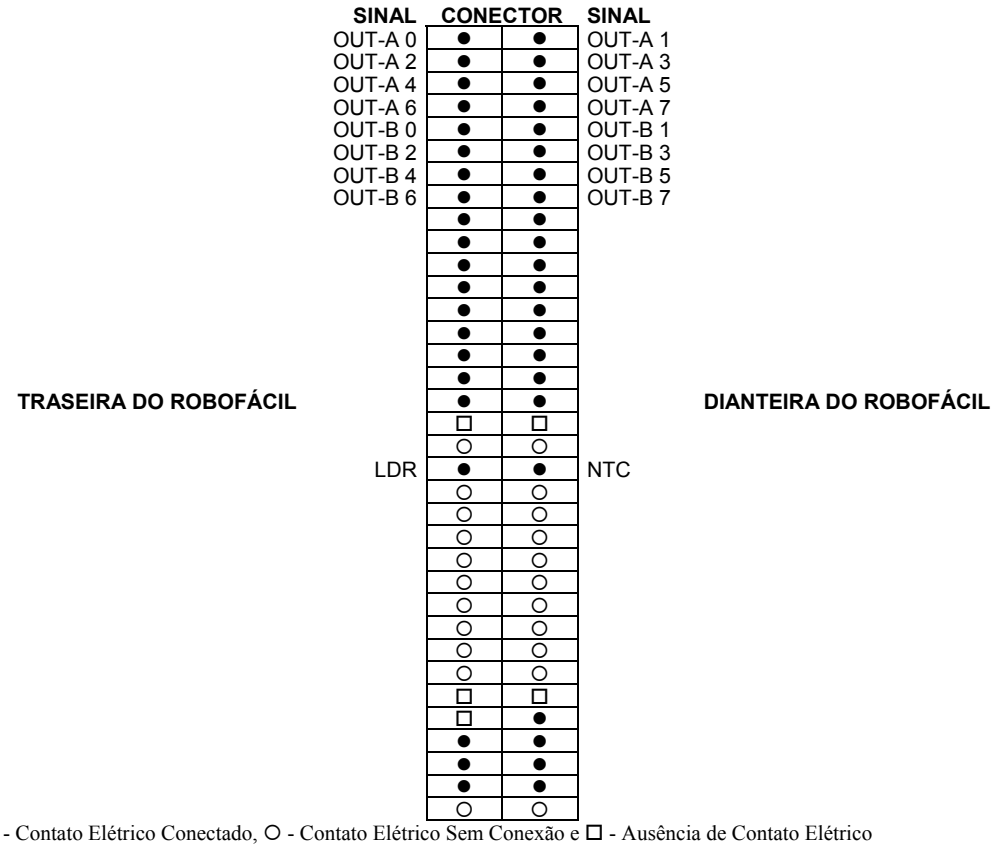
[8] RS-232 é um padrão da indústria para conexões de comunicação serial. Adotado pela Electrical Industries Association (EIA), este padrão recomendado – Recommended Standard (RS) – define as linhas características específicas e sinais utilizados por controladores de comunicação serial para padronizar a transmissão de dados seriais entre equipamentos.

acionamento desses motores. Numa versão industrializada do RoboFácil, essa fonte deverá ser suprimida.

3.7 – CONECTOR EXTERNO

Com o intuito de facilitar o interfaceamento entre a placa principal e os plugins, adotou-se um cabo *flat* de 34 vias (vide fig. 7) para realizar a ligação do conector da placa principal (vide fig. 8) – que contêm algumas entradas e saídas lógicas – com os *plugins* de controle dos motores de passo e leds[9]. Para realizar a ligação dos *plugins* de controle do sensor de temperatura e luminosidade, optou-se por utilizar apenas um fio para cada *plugin*, ligando-se o mesmo através de um conector fêmea em uma das entradas do conector macho, disponível na placa principal do RoboFácil (vide fig. 8).

O mapeamento apresentado abaixo especifica algumas saídas lógicas da placa principal.



Legenda: ● - Contato Elétrico Conectado, ○ - Contato Elétrico Sem Conexão e □ - Ausência de Contato Elétrico

Na fig. 8, pode-se observar o conector mapeado acima em detalhes.

[9] Pelo fato de existir a possibilidade, de na próxima versão do Kit se utilizar os outros sinais elétricos disponíveis no RoboFácil, não foram cortadas as vias, atualmente, não utilizadas do cabo *flat*.

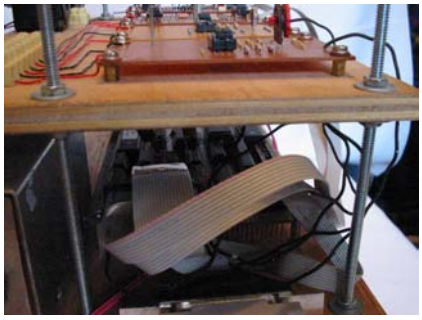


Figura 7 – Cabo flat e fios ligando o conector externo aos plugins

Figura 8 – Conector externo para ligação dos plugins

4 – DESENVOLVIMENTO DO SOFTWARE

4.1 – DEFININDO A LINGUAGEM DE PROGRAMAÇÃO

Visando o aprimoramento do Kit de Robótica Educacional, principalmente no que concerne à facilidade de utilização dessa ferramenta pedagógica em aplicações educacionais, propõem-se desenvolver uma linguagem icônica que tornará o RoboFácil muito mais amigável de ser programado por alunos do ensino fundamental e médio.

Nesse contexto – e para criar alicerces necessários a esse desenvolvimento – adotou-se, nessa etapa do projeto, um ambiente de desenvolvimento na linguagem C que gerasse código binário para o microcontrolador utilizado no RoboFácil. Após a realização de pesquisas e diversos testes com alguns produtos, optou-se por utilizar a ferramenta de desenvolvimento denominado µVision 2 v.2.37 [20].

Tal ferramenta possui muitas dos recursos disponíveis nos atuais ambientes de desenvolvimento RAD[10] de software, tais como o Borland® Delphi e o Microsoft® Visual Basic. Dentre eles podemos citar: *Debug*, *Breakpoint*, *Performance Analyzer*, *Memory Map*, *Inline assembly*, *Disassembly Windows*, etc.

5 – UTILIZANDO O ROBOFÁCIL

Atualmente, para que um usuário-aluno possa construir um programa para controlar o RoboFácil e fazê-lo executar determinadas tarefas é necessário seguir os seguintes passos: (1) o aluno desenvolve o programa com as funções que se deseja que o Kit realize através de programação utilizando a linguagem C, e fazendo uso das bibliotecas do *software* básico já

[10] RAD (Rapid Application Development) é uma sigla muito utilizada para ferramentas de desenvolvimento que tem recursos que facilitam o desenvolvimento de aplicações visuais.

implementadas (vide Anexo II); (2) compila o programa na ferramenta de desenvolvimento μVision 2 (vide item 4.1); (3) conecta o RoboFácil a uma *interface* serial do microcomputador; (4) inicializa o *software* HyperTerminal[11] e procede-se a transferência dos dados; (5) depois que o comportamento (programa do usuário-aluno) estiver carregado no Kit, deve-se comandar sua execução, bastando digitar "/8000" no HyperTerminal.

6 – CONCLUSÕES E TRABALHOS FUTUROS

Desde 2001, o Kit de Robótica Educacional do GINAPE vem sendo aperfeiçoado. No momento, o Kit apresenta-se numa versão protótipo montado sobre três bandejas. Na primeira, estão instaladas as rodas, na segunda a placa que contém os circuitos eletrônicos principais do RoboFácil e na terceira, os quatro *plugins* (controle de motores de passo, controle de leds, controle do sensor de temperatura e controle do sensor de luminosidade).

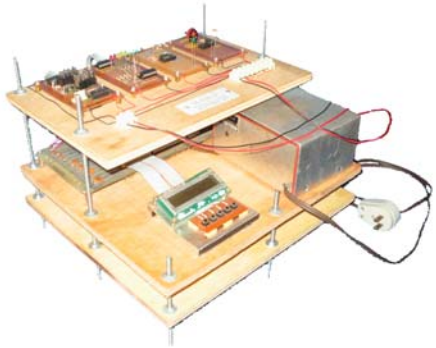


Figura 9 – Versão atual do RoboFácil

A arquitetura de *hardware* e *software* já implementada, permite a sua utilização em escolas de ensino médio profissionalizante de eletrônica, eletrotécnica, telecomunicações e informática. Entretanto, a inexistência de um ambiente de programação amigável para alunos de ensino fundamental e médio (linguagem icônica), ainda dificulta a sua utilização nestes espaços. A concepção e implementação de tal ambiente, faz parte do projeto de dissertação de Mestrado do 1º. autor deste trabalho.

Assim, como próximas atividades de aprimoramento do projeto temos:

- Desenvolvimento de um interpretador de comandos para facilitar a construção da linguagem icônica;
- Gravação em EPROM desse interpretador com as bibliotecas que controlam todos os recursos do RoboFácil;
- Implementar um algoritmo para controlar concorrência durante a utilização do Kit de Robótica. Esse desenvolvimento é imprescindível para que o Kit possa executar

[11] O HyperTerminal é um *software* que acompanha o Sistema Operacional Microsoft® Windows 95 e posteriores, muito utilizado para realizar comunicação serial, seja através de modem ou cabo serial ligando dois microcomputadores ou qualquer outro dispositivo serial, tais como, agendas eletrônicas, palmtops e o próprio RoboFácil.

operações de forma “simultânea” como, por exemplo, monitorar a luminosidade no sensor de luz enquanto realiza movimentos nos motores de passo e pisca alguns leds;

- Implementar uma fonte bivolt com o intuito de alimentar os circuitos eletrônicos do RoboFácil, incluindo os seus motores de passo. Essa fonte deverá gerar tensões de 5V e 12V com corrente de 1A e 3A, respectivamente;
- Implementar o controle da histeresse nos *plugins* do sensor de temperatura e luminosidade através de circuitos eletrônicos, tirando essa “responsabilidade” do *software*;
- Desenvolver em conjunto com a linguagem icônica uma *interface* amigável para enviar os comandos para o RoboFácil, como alternativa ao programa HyperTerminal, hoje utilizado;
- Utilizar o kit em ambientes reais de ensino para testar a sua funcionalidade e desenvolver material pedagógico para apoiar o trabalho do professor em sala de aula.

ANEXO I – ESQUEMAS ELETRÔNICOS DO ROBOFÁCIL

I.1 – CIRCUITO ELETRÔNICO PRINCIPAL (ROBOFAC.M51)

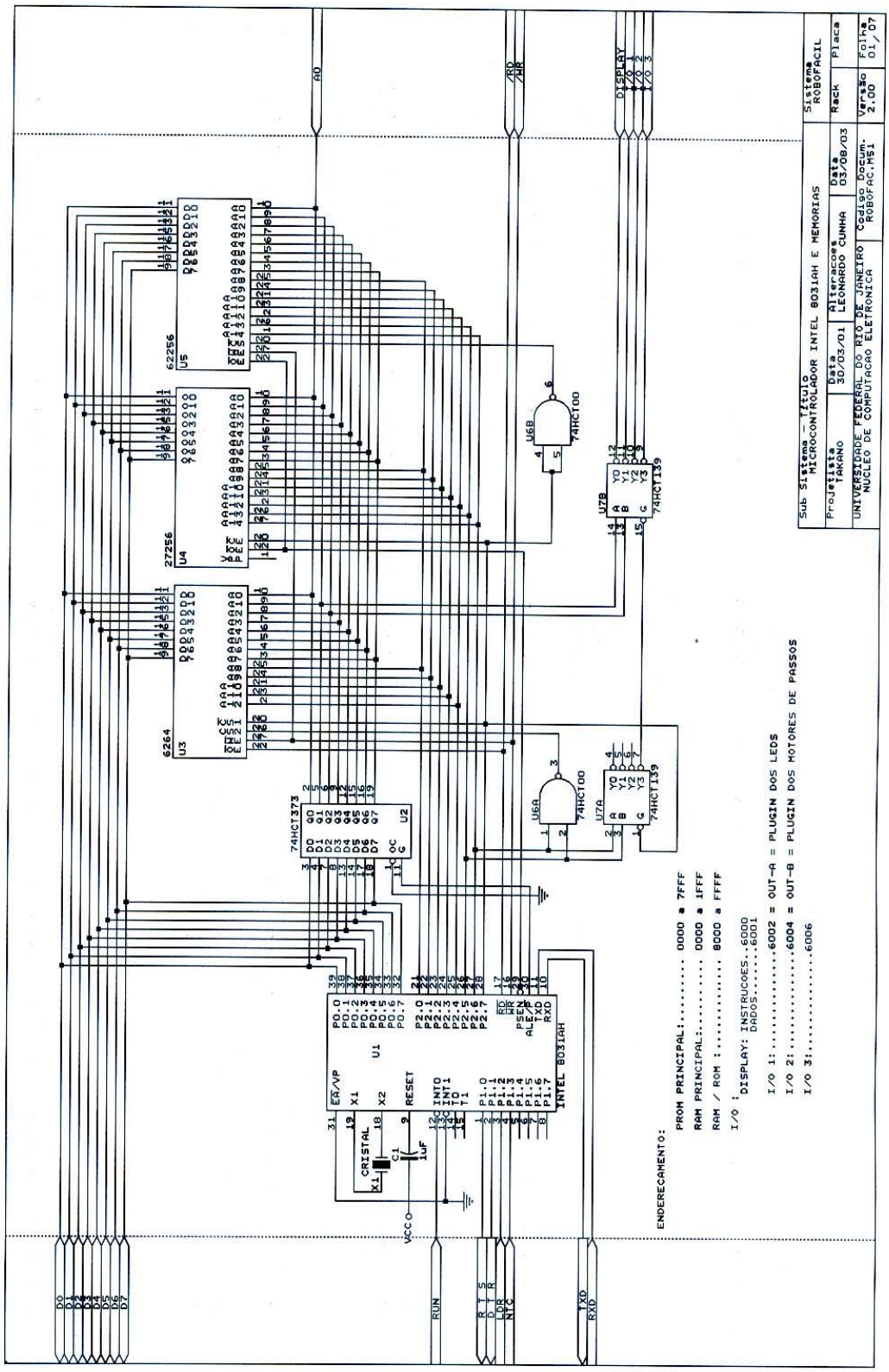


Fig. 10 – Esquema eletrônico da placa principal

I.2 – DISPLAY E PORTAS PARALELAS (ROBOFAC.IO)

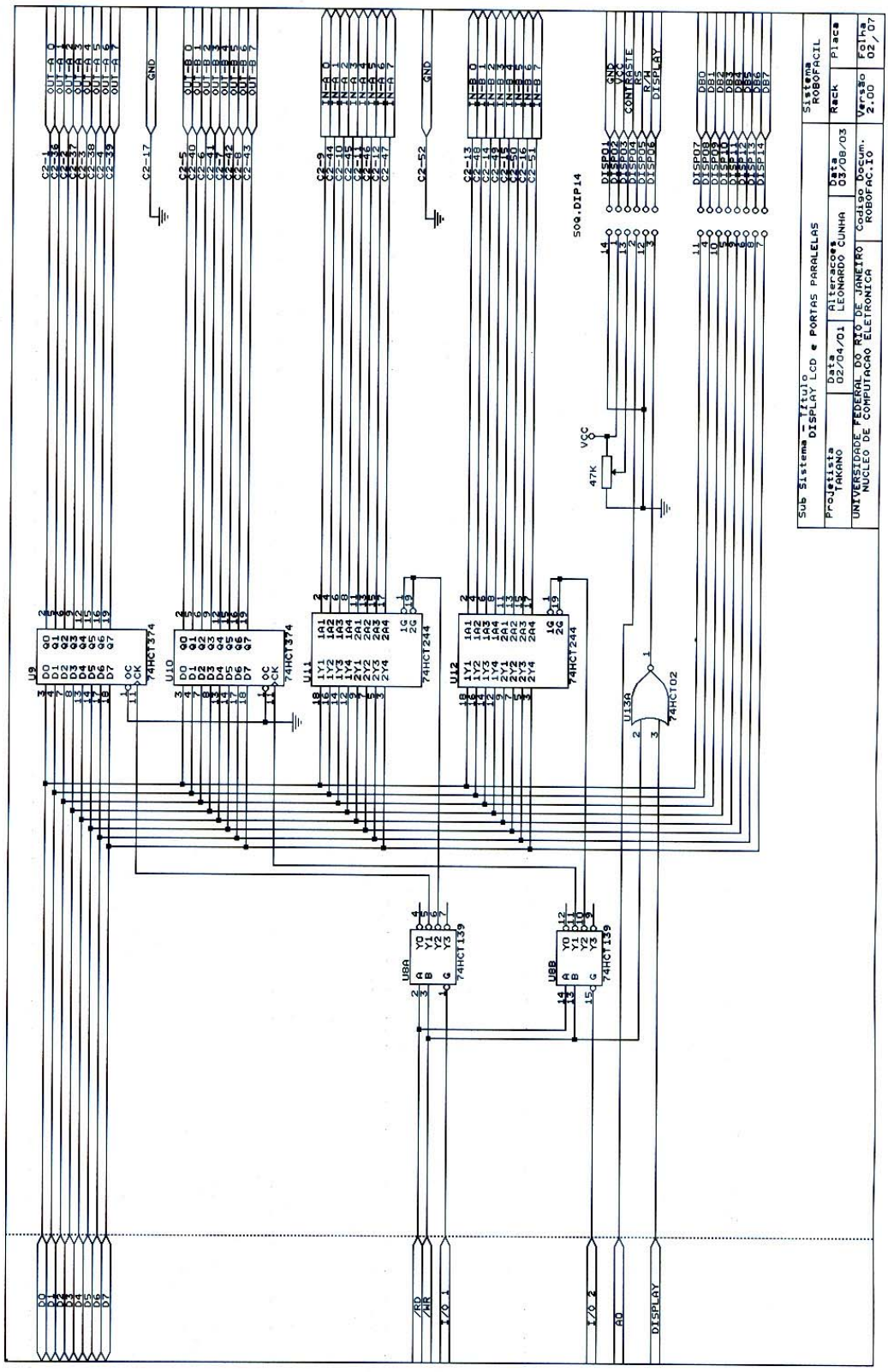


Fig. 11 – Esquema eletrônico do display e portas paralelas

I.3 – PORTA SERIAL E TECLADO (ROBOFAC.LIK)

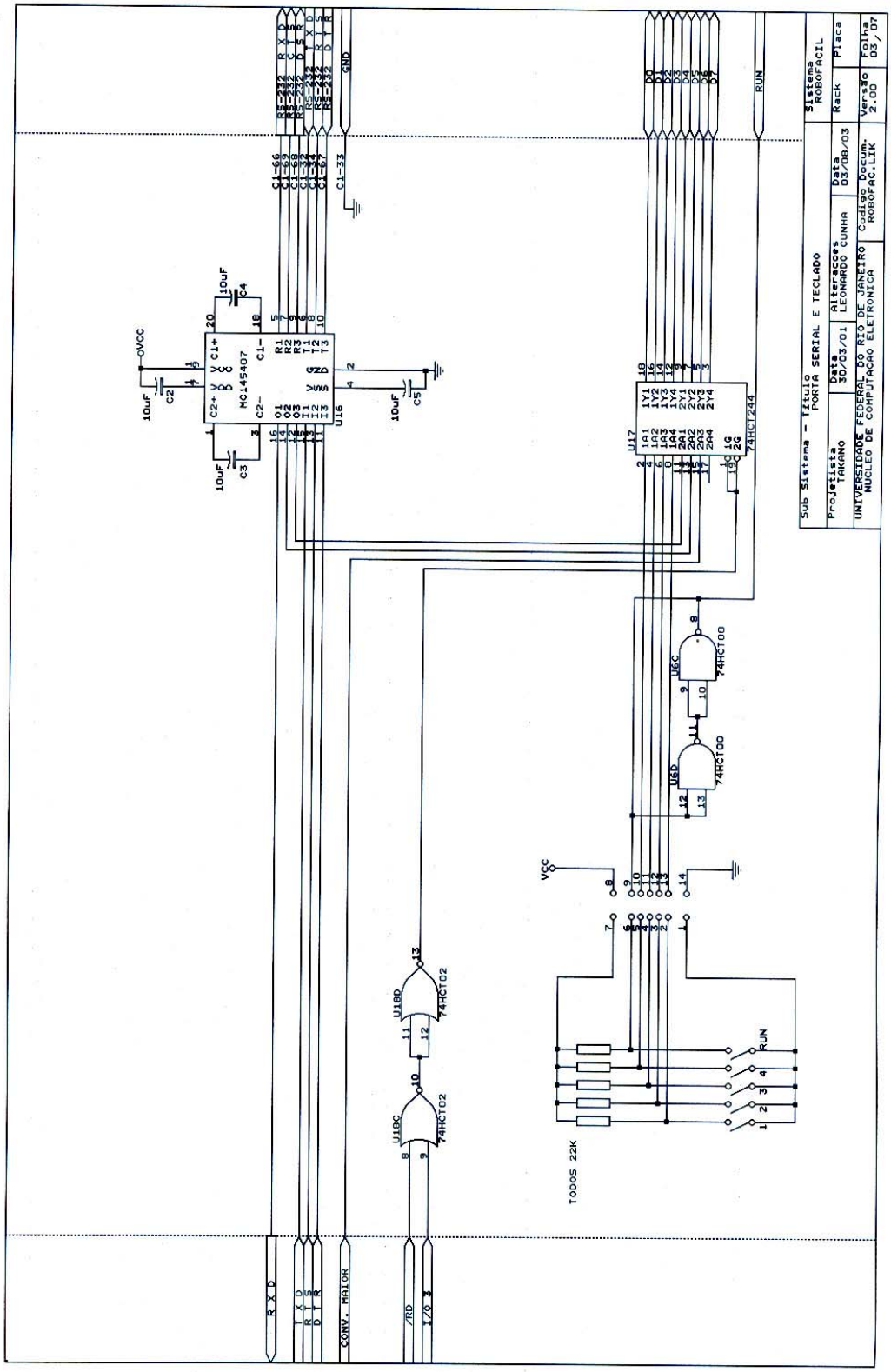


Fig. 12 – Esquema eletrônico da porta serial e teclado

I.4 – CONVERSORES DAC E ADC (ROBOFAC.DAC)

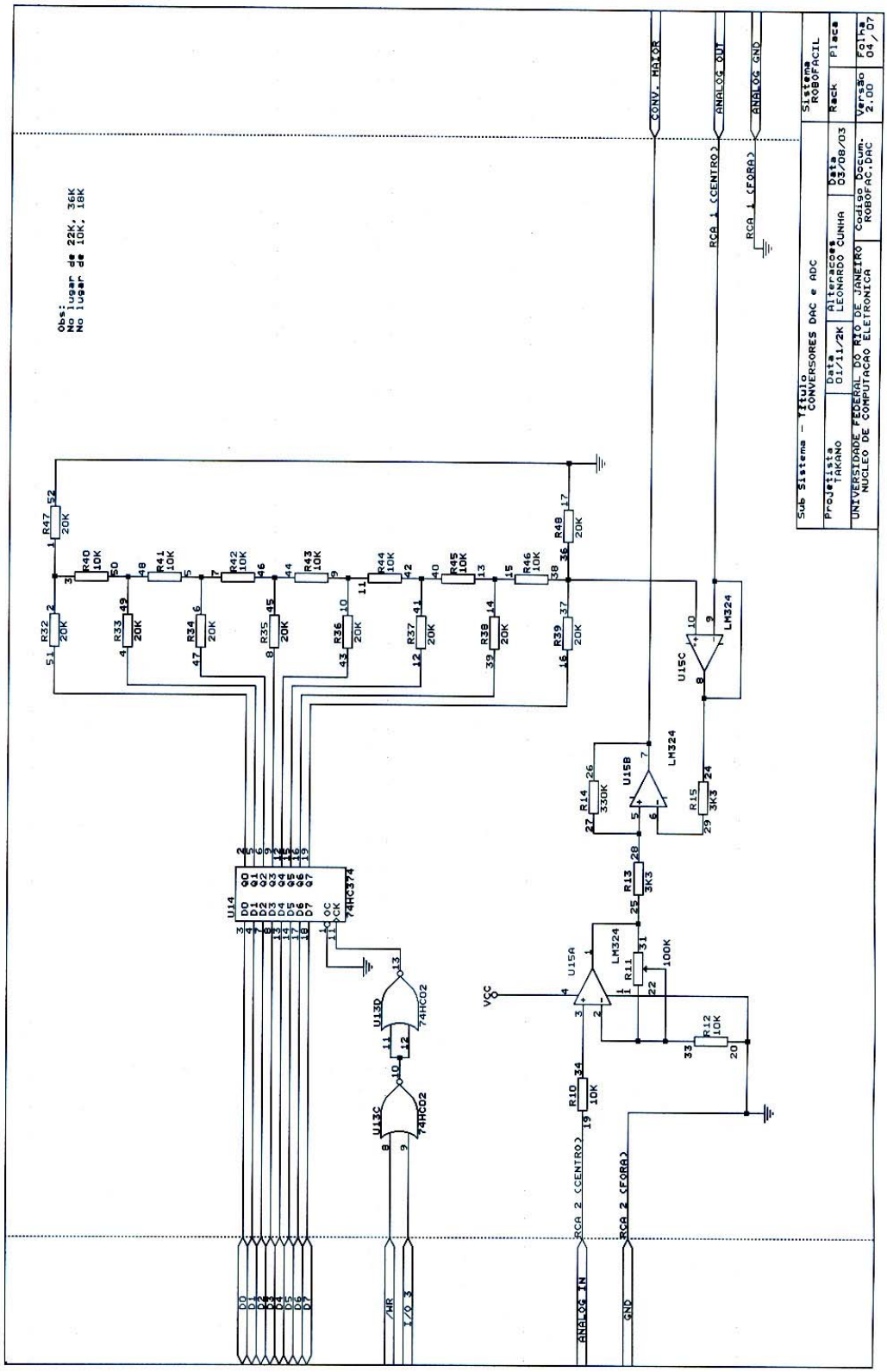


Fig. 13 – Esquema eletrônico dos conversores DAC e ADC

I.5 – PLUGIN DE CONTROLE DOS MOTORES DE PASSO E SENSORES (ROBOFAC.PMS)

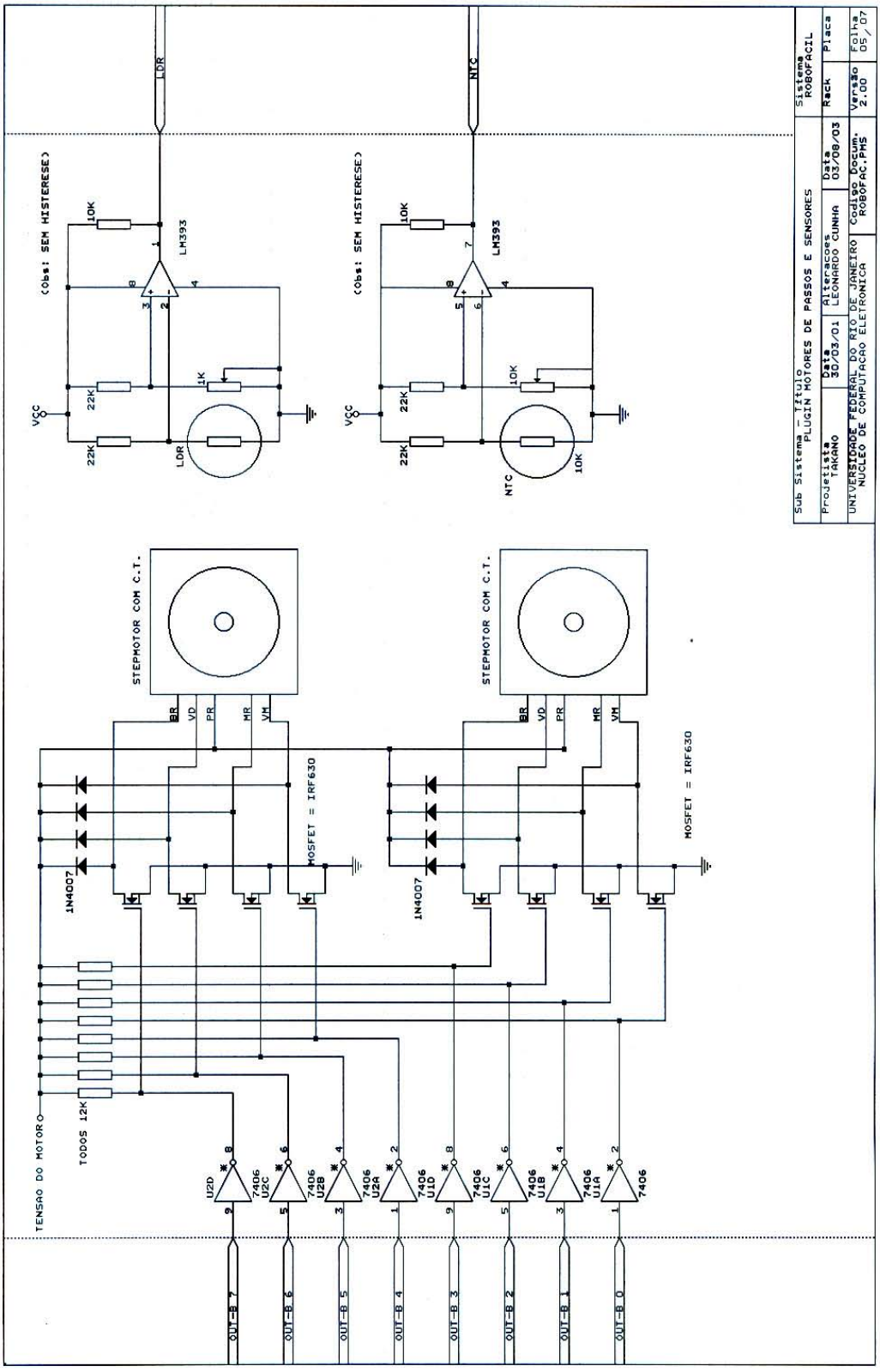


Fig. 14 – Esquema eletrônico dos motores de passo e sensores

I.6 – PLUGIN DE CONTROLE DOS LEDS E SENSORES (ROBOFAC.PLS)

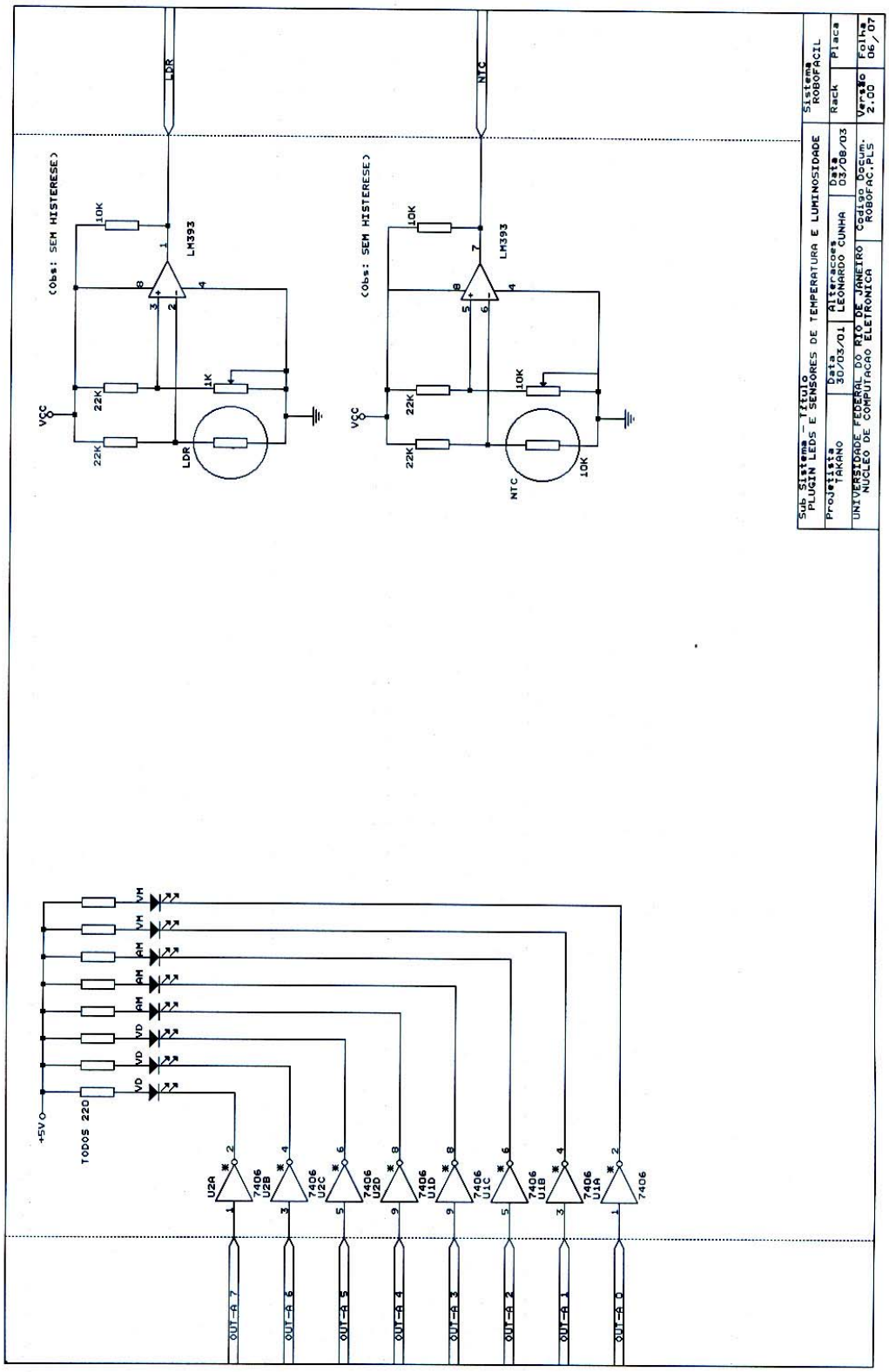


Fig. 15 – Esquema eletrônico dos leds e sensores

I.7 – CABO SERIAL

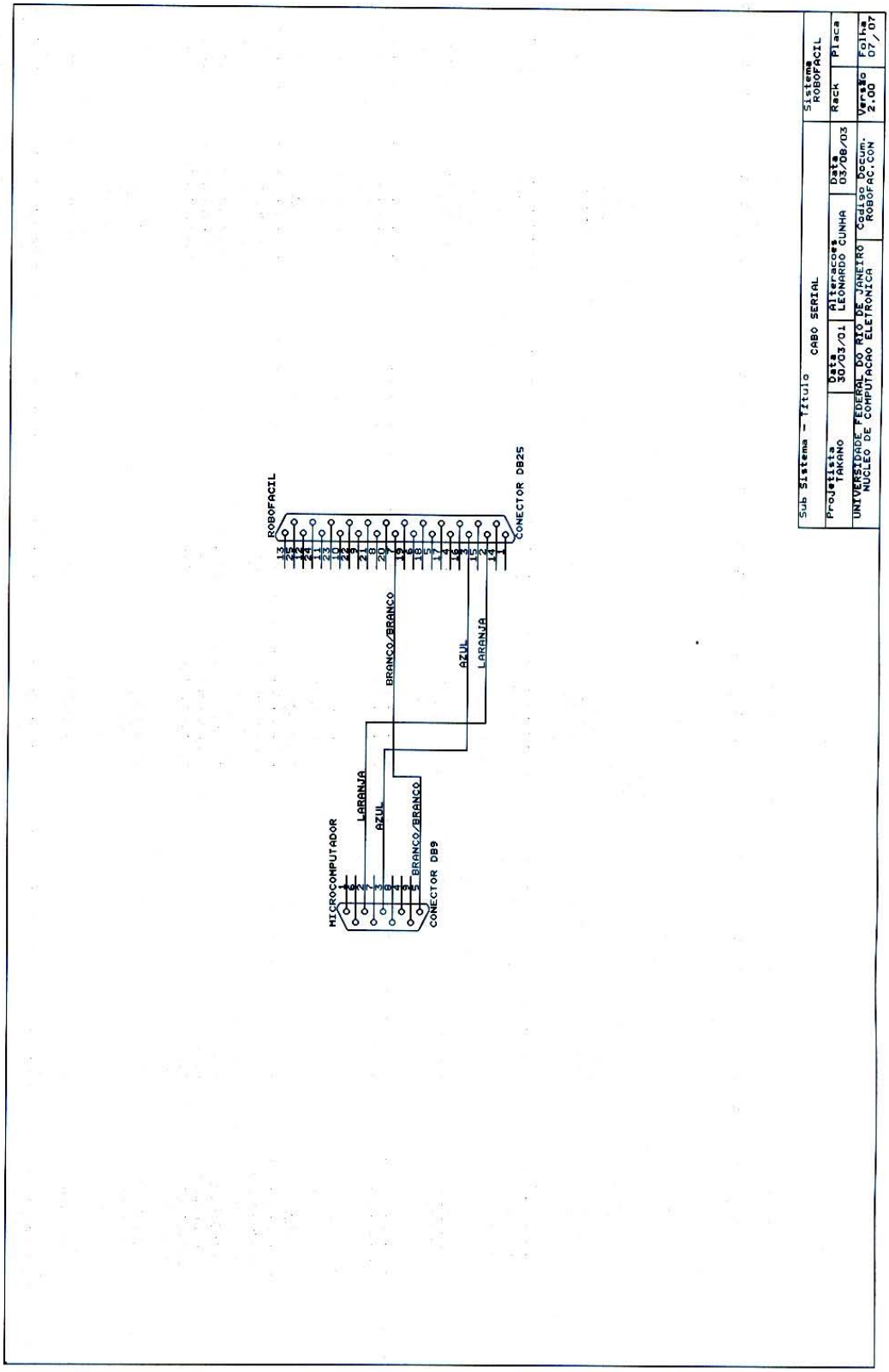


Fig. 16 – Ligação elétrica do cabo serial

ANEXO II – CÓDIGO-FONTE DO SOFWARE BÁSICO

O código-fonte implementado na linguagem C que controla o Kit de Robótica Educacional possui funções projetadas com o intuito de facilitar o aperfeiçoamento do RoboFácil, controlando o display (lcd.c), sensor de luminosidade (ldr.c), oito leds coloridos (led.c), motores de passo (motor.c), sensor de temperatura (ntc.c), comunicação pela *interface* serial do microcomputador com o Kit de Robótica (serial.c) e o tempo (timer.c). Salienta-se que a programação do *software* básico tem uma ligação intrínseca com o *hardware*, por esse motivo quando houver alterações eletrônicas, o *software* básico também deverá ser modificado.

II.1 – LCD.C

Para se ter o controle do *display* (como inicialização, limpeza e escrita de caracteres) foram desenvolvidas algumas funções.

```
/*
	Esse arquivo possui todas as funções para se utilizar
	o LCD.

	Orientador Acadêmico: José Antonio dos Santos Borges e
	                      Fábio Ferrentini Sampaio

	Autor: Leonardo Cunha de Miranda

	Primeira versão: 25/MAI/2003
	Última alteração: 30/JUN/2003
*/

#define INSTRUCAO_LCD 0x6000
#define DADO_LCD      0x6001
#define XBYTE ((unsigned char volatile xdata*) 0)

void PutInstructionLCD(char i) {
	int j;
	XBYTE [INSTRUCAO_LCD] = i;
	for (j=1;j!=100;j++);
}

void PutCharLCD(char c) {
	int i;
	XBYTE [DADO_LCD] = c;
	for (i=1;i!=100;i++);
}

void PutStringLCD(char *s) {
	do
		PutCharLCD(*s);
	while (*++s);
}

void InicializarLCD() {
	PutInstructionLCD(0x38);
	PutInstructionLCD(0x38);
	PutInstructionLCD(0x06);
	PutInstructionLCD(0x0C);
	PutInstructionLCD(0x01);
}

void ClearLCD() {
```

```
        PutInstructionLCD(0x01);
}
```

II.2 – LDR.C

Nesse arquivo foram implementadas as funções para se controlar o *plugin* do sensor de luminosidade (Light Dependent Resistor – LDR), através de uma função que verifica o *status* do sensor.

```
/*
        Esse arquivo possui todas as funções para se utilizar
        o LDR.

        Orientador Acadêmico: José Antonio dos Santos Borges e
                              Fábio Ferrentini Sampaio

        Autor: Leonardo Cunha de Miranda

        Primeira versão: 08/JUL/2003
        Última alteração: 03/AGO/2003
*/

char StatusLDR() {
        if (LDR==0x01) // 0x00 = Sem Luz, 0x01 ou ' ' = Com Luz
                return '1';
        else
                return '0';
}
```

II.3 – LED.C

Nesse arquivo foram desenvolvidas diversas funções para se controlar o *plugin* de *leds* que foi implementado no Kit de Robótica. Note que foram escritas funções para apagar e acender todos os oito *leds*, acender um *led* específico e acender todos os *leds* de uma das cores disponíveis – vermelho, amarelo e verde.

```
/*
        Esse arquivo possui todas as funções para se utilizar
        os LEDs.

        Orientador Acadêmico: José Antonio dos Santos Borges e
                              Fábio Ferrentini Sampaio

        Autor: Leonardo Cunha de Miranda

        Primeira versão: 13/JUL/2003
        Última alteração: 03/AGO/2003
*/

#define LED 0x6002
#define XBYTE ((unsigned char volatile xdata*) 0)

void PutDataLED(char c) {
        XBYTE [LED] = c;
}

void TurnOffAllLED() {
        PutDataLED(0x00);
```

```
}

void TurnOnAllLED() {
	PutDataLED(0xFF);
}

void TurnOnLED(int l) {
	switch (l) {
		case 1: PutDataLED(0x01); break; // Vermelho
		case 2: PutDataLED(0x02); break; // Vermelho
		case 3: PutDataLED(0x04); break; // Amarelo
		case 4: PutDataLED(0x08); break; // Amarelo
		case 5: PutDataLED(0x10); break; // Amarelo
		case 6: PutDataLED(0x20); break; // Verde
		case 7: PutDataLED(0x40); break; // Verde
		case 8: PutDataLED(0x80); break; // Verde
	}
}

void TurnOnRedLED() {
	PutDataLED(0x03);
}

void TurnOnYellowLED() {
	PutDataLED(0x1C);
}

void TurnOnGreenLED() {
	PutDataLED(0xE0);
}
```

II.4 – MAIN.C

O “programa principal” do RoboFácil foi modularizado com o objetivo de facilitar o acréscimo de outras funcionalidades ao Kit, por esse motivo esse arquivo referencia todas as outras bibliotecas que controlam o Kit de Robótica (lcd.c, ldr.c, led.c, motor.c, ntc.c, serial.c e timer.c). O código que está sendo apresentado abaixo foi o programa implementado que serviu como exemplo para apresentação aos meus orientadores Prof. Fábio Ferrentini Sampaio e José Antonio dos Santos Borges, exemplificando sua utilização.

```
/*
	Esse arquivo contém o programa principal do projeto de
	robótica educacional: RoboFácil.

	Orientador Acadêmico: José Antonio dos Santos Borges e
	                      Fábio Ferrentini Sampaio

	Autor: Leonardo Cunha de Miranda

	Primeira versão: 25/MAI/2003
	Última alteração: 03/AGO/2003
*/

#include "serial.c"
#include "lcd.c"
#include "ldr.c"
#include "ntc.c"
#include "led.c"
#include "motor.c"

main() {
	int i,j,Estado;
	MiliSeconds(100);
	InicializarSerial();
	MiliSeconds(100);
```

```
        InicializarLCD();
        MiliSeconds(100);
        TurnOffAllLED();
        PutStringSerial("\r\nRoboFácil\r\n\n");
        PutStringLCD("   RoboF");
        PutInstructionLCD(0xC0);
        PutStringLCD("ácil");
        PutStringSerial("Olá! Sou o RoboFácil.\r\n");
        PutStringSerial("Em cinco segundos começarei a realizar o que fui programado.\r\nObserve o
kit.\r\n\n");
        Seconds(5);
        for (i=1;i<=5;i++) {
                TurnOnAllLED();
                MiliSeconds(500);
                TurnOffAllLED();
                MiliSeconds(500);
        }
        TurnOnRedLED();
        Seconds(2);
        TurnOnGreenLED();
        Seconds(1);
        TurnMotor(3,1,1);
        TurnOnYellowLED();
        Seconds(2);
        TurnOnRedLED();
        Seconds(3);
        TurnOnGreenLED();
        Seconds(1);
        TurnMotor(3,0,1);
        for (i=1;i<=5;i++) {
                TurnOnYellowLED();
                MiliSeconds(500);
                TurnOffAllLED();
                MiliSeconds(500);
        }
        ClearLCD();
        PutStringLCD("Esperand");
        PutInstructionLCD(0xC0);
        PutStringLCD("o Calor?");
        Estado = 1;
        while (Estado) {
                if (StatusNTC() == '1')
                        Estado = 0;
        }
        ClearLCD();
        for (i=1;i<=5;i++) {
                for (j=1;j<=8;j++) {
                        TurnOnLED(j);
                        MiliSeconds(150);
                }
        }
        Seconds(1);
        for (i=1;i<=5;i++) {
                for (j=8;j>=1;j--) {
                        TurnOnLED(j);
                        MiliSeconds(300);
                }
        }
        for (i=1;i<=5;i++) {
                TurnOnYellowLED();
                MiliSeconds(500);
                TurnOffAllLED();
                MiliSeconds(500);
        }
        PutStringLCD("Esperand");
        PutInstructionLCD(0xC0);
        PutStringLCD("o Luz?");
        Estado = 1;
        while (Estado) {
                if (StatusLDR() == '1')
                        Estado = 0;
        }
        ClearLCD();
        PutStringLCD("   RoboF");
        PutInstructionLCD(0xC0);
        PutStringLCD("ácil");
        TurnMotor(3,1,5);
        Seconds(10);
```

```
        TurnMotor(3,0,5);
        for (i=1;i<=30;i++) {
                TurnOnAllLED();
                MiliSeconds(250);
                TurnOffAllLED();
                MiliSeconds(250);
        }
        PutStringSerial("Tchau!\r\n\n");
}
```

II.5 – MOTOR.C

Nesse arquivo existem implementações de várias funções para se controlar os motores de passo utilizados no Kit de Robótica Educacional. Ressalta-se que já foram desenvolvidas funções para controlar a direção – sentido horário e anti-horário, a velocidade de rotação e o número de voltas completas dos motores. Todavia, se necessário, as funções mais primitivas podem realizar meia volta e até um quarto de volta sobre o eixo dos motores.

```
/*
        Esse arquivo possui todas as funções para se utilizar
        os motores.

        Orientador Acadêmico: José Antonio dos Santos Borges e
                              Fábio Ferrentini Sampaio

        Autor: Leonardo Cunha de Miranda

        Primeira versão: 13/JUL/2003
        Última alteração: 03/AGO/2003
*/

#include "timer.c"

#define MOTORES 0x6004
#define XBYTE ((unsigned char volatile xdata*) 0)

void PutDataMotor(char c,int v) {
        MiliSeconds(5*v);
        XBYTE [MOTORES] = c;
}

void HalfStepMotor(int d,v) {
        switch (d) {
                case 0: PutDataMotor(0x88,v); // Sentido Anti-Horário
                                PutDataMotor(0xCC,v);
                                PutDataMotor(0x22,v);
                                PutDataMotor(0x33,v);
                                PutDataMotor(0x11,v);
                                PutDataMotor(0x99,v);
                                PutDataMotor(0x44,v);
                                PutDataMotor(0x66,v);
                                break;
                case 1: PutDataMotor(0x88,v); // Sentido Horário
                                PutDataMotor(0x66,v);
                                PutDataMotor(0x44,v);
                                PutDataMotor(0x99,v);
                                PutDataMotor(0x11,v);
                                PutDataMotor(0x33,v);
                                PutDataMotor(0x22,v);
                                PutDataMotor(0xCC,v);
                                break;
        }
}

void OneStepMotor(int d,v) {
        int i;
```

```
        for (i=1;i<=2;i++)
                HalfStepMotor(d,v);
}

void QuaterTurnMotor(int d,v) {
        int i;
        for (i=1;i<=3;i++)
                OneStepMotor(d,v);
}

void HalfTurnMotor(int d,v) {
        int i;
        for (i=1;i<=2;i++)
                QuaterTurnMotor(d,v);
}

void OneTurnMotor(int d,v) {
        int i;
        for (i=1;i<=2;i++)
                HalfTurnMotor(d,v);
        HalfStepMotor(d,v);
}

void TurnMotor(int t,d,v) { // t = Número de Voltas, d = Sentido da(s) Volta(s)
        int i;                  // d = 1 (Sentido Horário), d = 0 (Sentido Anti-Horário)
        for (i=1;i<=t;i++)      // v = Velocidade = 5ms x v = Quanto menor v mais rápido é o motor
                OneTurnMotor(d,v);
}
```

II.6 – NTC.C

Para se controlar o *plugin* do sensor de temperatura (Negative Temperature Coefficient – NTC), implementou-se uma função que verifica o *status* do sensor.

```
/*
        Esse arquivo possui todas as funções para se utilizar
        o NTC.

        Orientador Acadêmico: José Antonio dos Santos Borges e
                              Fábio Ferrentini Sampaio

        Autor: Leonardo Cunha de Miranda

        Primeira versão: 10/JUL/2003
        Última alteração: 30/JUL/2003
*/

char StatusNTC() {
        if (NTC==0x01) // 0x00 = Sem Calor, 0x01 ou ' ' = Com Calor
                return '1';
        else
                return '0';
}
```

II.7 – REG52.H

Nesse arquivo são definidos os Registros de Funções Especiais (Special Function Registers – SFR). Aproveitou-se uma biblioteca padrão que é disponibilizada pelo ambiente de desenvolvimento e acrescentou-se alguns bits especiais de controle que viabilizaram o aprimoramento do *software* básica do RoboFácil.

```
/*------------------------------------------------------------
REG52.H

Header file for generic 80C52 and 80C32 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
------------------------------------------------------------*/

#ifndef __REG52_H__
#define __REG52_H__

/*  BYTE Registers  */
sfr P0    = 0x80;
sfr P1    = 0x90;
sfr P2    = 0xA0;
sfr P3    = 0xB0;
sfr PSW   = 0xD0;
sfr ACC   = 0xE0;
sfr B     = 0xF0;
sfr SP    = 0x81;
sfr DPL   = 0x82;
sfr DPH   = 0x83;
sfr PCON  = 0x87;
sfr TCON  = 0x88;
sfr TMOD  = 0x89;
sfr TL0   = 0x8A;
sfr TL1   = 0x8B;
sfr TH0   = 0x8C;
sfr TH1   = 0x8D;
sfr IE    = 0xA8;
sfr IP    = 0xB8;
sfr SCON  = 0x98;
sfr SBUF  = 0x99;

/*  8052 Extensions  */
sfr T2CON  = 0xC8;
sfr RCAP2L = 0xCA;
sfr RCAP2H = 0xCB;
sfr TL2    = 0xCC;
sfr TH2    = 0xCD;

/*  BIT Registers  */
/*  PSW  */
sbit CY    = PSW^7;
sbit AC    = PSW^6;
sbit F0    = PSW^5;
sbit RS1   = PSW^4;
sbit RS0   = PSW^3;
sbit OV    = PSW^2;
sbit P     = PSW^0; //8052 only

/*  TCON  */
sbit TF1   = TCON^7;
sbit TR1   = TCON^6;
sbit TF0   = TCON^5;
sbit TR0   = TCON^4;
sbit IE1   = TCON^3;
sbit IT1   = TCON^2;
sbit IE0   = TCON^1;
sbit IT0   = TCON^0;

/*  IE  */
sbit EA    = IE^7;
sbit ET2   = IE^5; //8052 only
sbit ES    = IE^4;
sbit ET1   = IE^3;
sbit EX1   = IE^2;
sbit ET0   = IE^1;
sbit EX0   = IE^0;

/*  IP  */
sbit PT2   = IP^5;
sbit PS    = IP^4;
sbit PT1   = IP^3;
sbit PX1   = IP^2;
sbit PT0   = IP^1;
```

```
sbit PX0   = IP^0;

/*  P3  */
sbit RD    = P3^7;
sbit WR    = P3^6;
sbit T1    = P3^5;
sbit T0    = P3^4;
sbit INT1  = P3^3;
sbit INT0  = P3^2;
sbit TXD   = P3^1;
sbit RXD   = P3^0;

/*  SCON  */
sbit SM0   = SCON^7;
sbit SM1   = SCON^6;
sbit SM2   = SCON^5;
sbit REN   = SCON^4;
sbit TB8   = SCON^3;
sbit RB8   = SCON^2;
sbit TI    = SCON^1;
sbit RI    = SCON^0;

/*  P1  */
sbit NTC   = P1^3; // Bit de controle do NTC
sbit LDR   = P1^2; // Bit de controle do LDR
sbit T2EX  = P1^1; // 8052 only
sbit T2    = P1^0; // 8052 only

/*  T2CON  */
sbit TF2    = T2CON^7;
sbit EXF2   = T2CON^6;
sbit RCLK   = T2CON^5;
sbit TCLK   = T2CON^4;
sbit EXEN2  = T2CON^3;
sbit TR2    = T2CON^2;
sbit C_T2   = T2CON^1;
sbit CP_RL2 = T2CON^0;

#endif
```

II.8 – SERIAL.C

Nesse arquivo implementou-se diversas funções para controlar a comunicação entre o Kit de Robótica Educacional e o microcomputador, através da *interface* serial de ambos os dispositivos. É através dessa *interface* que o RoboFácil recebe o "comportamento" que o usuário gostaria que o Kit realize.

```
/*
       Esse arquivo possui todas as funções para se utilizar
       a interface serial.

       Orientador Acadêmico: José Antonio dos Santos Borges e
                             Fábio Ferrentini Sampaio

       Autor: José Antonio dos Santos Borges e
              Leonardo Cunha de Miranda

       Primeira versão: 20/MAI/2003
       Última alteração: 05/JUN/2003
*/

#include <reg52.h>

InicializarSerial() {
       TH1   = 0xDD;
       TL1   = 0xDD;
       TMOD  = 0x20;
```

```
        SCON  = 0x52;
        TMOD  = 0x20;
        TCON  = 0x40;
        TI    = 1;
}

PutCharSerial (c)
char c;
{
        while (!TI)
                ;
        TI = 0;
        SBUF = c;
}

PutStringSerial (s)
char *s;
{
        while (*s)
                PutCharSerial(*s++);
}

char GetCharSerial() {
        while (!RI)
                ;
        RI = 0;
        return SBUF;
}

GetStringSerial(s)
char *s;
{
        char c;
        do {
                c = GetCharSerial();
                *s++ = c;
        } while (c != '\r');
        PutCharSerial ('\n');
}
```

II.9 TIMER.C

Nesse são expostas funções para controlar o tempo durante a utilização do RoboFácil. Note que já foram desenvolvidas funções para se esperar milisegundos, segundos e minutos.

```
/*
        Esse arquivo possui todas as funções para se controlar
        o tempo.

        Orientador Acadêmico: José Antonio dos Santos Borges e
                              Fábio Ferrentini Sampaio

        Autor: Leonardo Cunha de Miranda

        Primeira versão: 29/JUL/2003
        Última alteração: 03/AGO/2003
*/

void MiliSeconds(long int t) { // t = Tempo em milisegundos
        long int i;
        int j;
        for (i=1;i<=t;i++)
                for (j=1;j<=35;j++); // Conta um (01) milisegundos
}

void Seconds(int t) { // t = Tempo em segundos
        int i;
        for (i=1;i<=t;i++)
```

```
                MiliSeconds(1000); // Conta um (01) segundo
}

void Minutes(int t) { // t = Tempo em minutos
        int i;
        for (i=1;i<=t;i++)
                Seconds(60); // Conta um (01) minuto
}
```

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **535 Simulator**. Disponível em: <http://personales.mundivia.es/hvasquez/sim535>. Acesso em: 08 abr. 2003

[2] **A History of Education Robotics**. Disponível em: <http://www.edurobot.com>. Acesso em: 08 abr. 2003

[3] **A Study of Robotics in the Classroom**. Disponível em: <http://www.edurobot.com>. Acesso em: 08 abr. 2003

[4] **ARS Consult**. Disponível em: <http://www.ars.com.br>. Acesso em: 30 abr. 2003

[5] BARBACENA, Ilton L.; FLEURY, Cláudio Afonso. **Display LCD**. INTECH. out. 1996

[6] BORGES, J. A. S.; GANDRA, H. **Sistema para Desenvolvimento de Projetos Educacionais em Robótica**. 29 p. NCE/UFRJ. 2001

[7] **Construction Robotics Research Group/Computing Department/Lancaster University**. Disponível em: <http://www.comp.lancs.ac.uk/engineering/research/mechatronics/constrobotics/main.html>. Acesso em: 28 jun. 2003

[8] D'ABREU, J. V. V. **Desenvolvimento de Ambientes de Aprendizagem Baseados no Uso de Dispositivos Robóticos**. SBIE. 1999

[9] D'ABREU, J. V. V.; CHELLA, M. T. **Ambiente de Telerobótica em EaD**. Anais do XXIII Congresso da Sociedade Brasileira de Computação. ago. 2003. Volume V. Anais do IX Workshop sobre Informática na Escola (IX WIE)

[10] **DISPLAY'S LCD/¿Cómo usarlos?**. Disponível em: <http://www.pablin.com.ar/electron/info/lcd>. Acesso em: 16 mai. 2003

[11] FRÓES, J. R. M. **O Computador na Educação**. XV Congresso Nacional de Educação – AEC/BR. Oficina: Informática Educacional: Disciplina e Instrumento no Processo Educacional

[12] HOYLES, Celia. **Programming Rules: What Do Children Understand?** Institute of Education, University of London, UK

[13] KINOSHITA, Jorge; HIRAKAWA, André. **Linux: Driver para Display na Paralela**. Escola Politécnica da USP/Departamento de Engenharia de Computação e Sistemas Digitais – PCS

[14] KIRAKANA, André Riyuiti; CUGNASCA, Carlos Eduardo. **Linguagem de alto nível "c" para 8051 e programação estruturada**. Escola Politécnica da USP/Departamento de Engenharia de Computação e Sistemas Digitais – PCS. Disponível em: <http://www.pcs.usp.br/~pcs2497/2497e042002.pdf>. Acesso em: 16 mai. 2003

[15] **LEGO Lab**. Department of Computer Science (DAIMI)/Faculty of Science/University of Aarhus. Disponível em: <http://legolab.daimi.au.dk>. Acesso em: 28 jun. 2003

[16] **LEGO.com Mindstorms Home**. Disponível em: <http://mindstorms.lego.com>. Acesso em: 28 jun. 2003

[17] **LEGO.com Play On!**. Disponível em: <http://www.lego.com>. Acesso em: 28 jun. 2003

[18] LIMA, Alessandro de Souza; ROSA, Vagner Santos da. **Microcontrolador 8051**. Disponível em: <http://lula.dmat.furg.br/~vagner/8051emu/apostila>. Acesso em: 08 abr. 2003

[19] **Massachusetts Institute of Technology (MIT)**. Disponível em: <http://www.mit.edu>. Acesso em: 28 jun. 2003
[20] **µVision 2**. Disponível em: <http://www.keil.com>. Acesso em: 30 abr. 2003
[21] **Microcontroller Instruction Set**. ATMEL
[22] MIRANDA, L. C. **Desenvolvimento de Hardware e Software do Kit de Robótica Educacional RoboFácil**. 99 p. NCE/UFRJ. 2003
[23] MIRANDA, L. C.; BORGES, J. A. S.; SAMPAIO, F. F. **RoboFácil: Kit de Robótica para Uso Educacional**. Disponível em: <http://intervox.nce.ufrj.br/~leonardo>. Acesso em: 10 set. 2003
[24] **Módulos Multi-Matrix ALFACOM – Manual de utilização**. ALFATRONIC S.A.
[25] Power Field Effect Transistor. Motorola Semiconductor Technical Data. **Motorola TMOS Power Mosfet Data**. 127 p.
[26] **Projeto RoboFácil**. Disponível em: <http://www.leonardocunha.com.br/robofacil>. Acesso em: 27 jan. 2004
[27] **Revista Mecatrônica Atual**. Disponível em <http://www.mecatronicaatual.com.br>. Acesso em: 28 jul. 2003
[28] **Robotic Life**. Disponível em: <http://robotic.media.mit.edu>. Acesso em: 28 jun. 2003
[29] **RobotIcanDo**. Laboratório de Estudos Cognitivos da Universidade Federal do Rio Grande do Sul (LEC/UFRGS). Disponível em: <http://oea.psico.ufrgs.br/roboticando>. Acesso em: 28 jun. 2003
[30] SETZER, Waldemar W. **Computadores na Educação: Porquê, Quando e Como?**
[31] **Sistemas Robóticos com SuperLogo (SIROS)**. Equipe de Robótica Pedagógica do Núcleo de Informática Aplicada a Educação da Universidade de Campinas (NIED/UNICAMP). Disponível em: <http://www.nied.unicamp.br/~siros>. Acesso em: 28 jun. 2003
[32] **Super Robby**. ARS Consult. Disponível em: <http://www.ars.com.br/produtos/srobby>. Acesso em: 30 abr. 2003
[33] **The Robotics Institute/Carnegie Mellon University**. Disponível em: <http://www.ri.cmu.edu>. Acesso em: 28 jun. 2003